AMENDMENTS TO THE CLAIMS:

This listing of claims will replace all prior versions, and listings, of claims in the application:

LISTING OF CLAIMS:

1. (Currently Amended) A method for monitoring progress with the execution of a linear sequence of instructions in a computer program stored in a computer-readable memory, comprising the steps of analysing the sequence of instructions transmitted from said memory to a processor intended to execute the program being monitored by extracting a data item from each instruction transmitted to the processor and performing a calculation on said data item, and verifying the result of this analysis by comparing the result of said calculation to reference data, recorded with said program, wherein the reference data comprises a value pre-established so as to correspond to the result of the analysis produced during the monitoring method only if all the instructions in the sequence of instructions have actually been analysed during the running of the program.

2. (Previously Presented) A method according to Claim 1, wherein the verification of the result of the analysis is caused by an instruction placed at a predetermined location in the program to be monitored, said instruction containing the reference data relating to a set of instructions whose correct execution is to be monitored.

3. (Previously Presented) A method according to Claim 1 wherein, when the instructions of the set of instructions to be monitored are in the form of a value, said analysis of the instructions is carried out by using these instructions as a numerical value.

4. (Previously Presented) A method according to Claim 1, comprising the steps of:

- during the preparation of the program to be monitored:

  - incorporating, in at least one predetermined location in a sequence of instructions in the program, a reference value established according to a predetermined rule applied to identifiable data in each instruction to be monitored, and

- during the execution of the program to be monitored:

  - obtaining said identifiable data in each instruction received for execution,

  - applying said predetermined rule to said identifiable data thus obtained in order to establish a verification value, and

  - verifying that this verification value actually corresponds to the reference value recorded with the program.

5. (Previously Presented) A method according to Claim 1, further comprising a step of interrupting the flow of the program if the analysis reveals that the program being monitored has not been run as expected.

6. (Previously Presented) A method according to Claim 1, further comprising an invalidation step for future use of the device comprising the monitored program if said analysis reveals a predetermined number of times that the program being monitored has not run in the expected manner.

7. (Previously Presented) A method according to Claim 1 wherein the set of instructions to be monitored does not include jumps in its expected flow.

8. (Previously Presented) A method according to Claim 1 wherein, when the program to be monitored provides for at least one jump, the monitoring method is applied separately to sets of instructions in the program which do not include jumps between two successive instructions.

9. (Previously Presented) A method according to Claim 8, wherein, when the program to be monitored includes an instruction for a jump dependent on the manipulated data, the monitoring method is implemented separately for a set of instructions which precedes the jump, and for at least one set of instructions which follows said jump.

10. (Previously Presented) A method according to Claim 9, wherein, for a set of instructions providing for a jump, an instruction which controls this jump is integrated in said set of instructions for the purpose of obtaining a verification value for this set of instructions before executing the jump instruction.

11. (Previously Presented) A method according to Claim 1 wherein the analysis is reinitialised before each new monitoring of a sequence of instructions to be monitored.

12. (Previously Presented) A method according to Claim 11, wherein the reinitialisation of the analysis of each new monitoring includes the step of erasing or replacing a verification value obtained during a previous analysis.

13. (Previously Presented) A method according to Claim 11 wherein the reinitialisation of the monitoring analysis is controlled by the software itself.

14. (Previously Presented) A method according to Claim 1 wherein the analysis produces a verification value obtained as the last value in a series of values which is made to change successively with the analysis of each of the analysed instructions of the set of instructions, thus making it possible to contain an internal state of the running of the monitoring method and to follow its changes.

15. (Previously Presented) A method according to Claim 1 wherein the analysis includes the step of calculating, for each instruction under consideration following a previous instruction, the result of an operation on both a value obtained of the instruction in question and the result obtained by the same operation performed on the previous instruction.

16. (Previously Presented) A method according to Claim 1 wherein the analysis includes the step of recursively applying a hash function to values obtained of each monitored instruction, starting from a last initialisation performed.

17. (Previously Presented) A method according to Claim 1 wherein the analysis includes the step of making a verification value change by performing a redundancy calculation on all the operating codes and the addresses executed since the last initialisation was carried out.

18. (Previously Presented) A method according to Claim 1 wherein the analysis includes the step of obtaining a comparison value by calculating successive intermediate values as the data of the respective instructions are obtained.

19. (Previously Presented) A method according to Claim 1 wherein the analysis comprises a step of saving each data item necessary for verification, obtained from instructions in the set of instructions to be monitored as they are executed, and performing a calculation of a verification value from these data only at the necessary time, once all the necessary data have been obtained.

20. (Currently Amended) A device for monitoring progress with the execution of a series of instructions of a computer program stored in a computer-readable memory, comprising means for analysing the sequence of instructions transmitted from said memory to the processor intended to execute the program being monitored by extracting a data item from each instruction transmitted to the

processor and performing a calculation on said data item, and means for verifying the result of this analysis by comparing the result of said calculation to reference data recorded with said program, wherein the reference data comprises a value pre-established so as to correspond to the result of the analysis produced during monitoring only if all the instructions in the sequence of instructions have actually been analysed during the running of the program.

21. (Previously Presented) A device according to Claim 20, further including a register for recording intermediate results in a calculation in a chain carried out by the analysis means in order to obtain a verification value.

22. (Previously Presented) A device according to Claim 21, further comprising means for recording a predetermined value or resetting the register under the control of an instruction transmitted during the execution of a program to be monitored.

23. (Previously Presented) A device according to Claim 20, further comprising means for counting the number of unexpected events in the program being monitored, as determined by the analysis means, and means for invalidating the future use of the program to be monitored if this number reaches a predetermined threshold.

24. (Previously Presented) A device according to Claim 20 that is integrated into a programmed device containing said program to be monitored.

25. (Previously Presented) A device according to Claim 20 that is integrated into a program execution device.

26. (Currently Amended) A program execution device that executes a series of instructions of a computer program stored in a computer-readable memory, comprising means for analysing the sequence of instructions ~~transmitted~~ retrieved from said memory for execution by extracting a data item from each instruction and performing a calculation on said data item, and means for verifying the result of this analysis by comparing the result of said calculation to reference data recorded with the program to be monitored, wherein the reference data comprises a value pre-established so as to correspond to the result of the analysis produced during monitoring only if all the instructions in the sequence of instructions have actually been analysed during the running of the program.

27. (Cancelled)

28. (Currently Amended) A programmed device containing a computer-readable memory storing a program having a series of recorded instructions and a ~~fixed memory containing~~ also storing reference data pre-established as a function of data contained in said instructions for analysis and verification of the sequence of instructions, wherein the reference data comprises a value pre-established so as to correspond to the result of the analysis produced during monitoring only if all the

instructions in the sequence of instructions have actually been analyzed during the ~~running~~ execution of the program by said device.

29. (Previously Presented) A device according to Claim 28, wherein said device is a smart card.

30. (Previously Presented) A device according to Claim 28 wherein the reference data are recorded in the form of a prewired value or values fixed in memory.

31. (Previously Presented) A device for programming a programmed device according to Claim 28, comprising means for entering, in at least one predetermined location in a sequence of instructions in the program, a reference value calculated according to a pre-established mode from data included in each instruction in a set of instructions whose execution is to be monitored.

32. (Cancelled)

33. (Currently Amended) A method for verifying proper execution of a sequence of instructions in a program ~~by a processor~~ stored in a computer-readable memory, comprising the following steps:

(i) retrieving an instruction in said sequence from ~~an instruction register~~ said memory, for execution by the processor;

(ii) calculating an updated monitoring value by processing data contained in the retrieved instruction with a stored monitoring value, and replacing the stored monitoring value with the updated monitoring value;

(iii) loading the retrieved instruction into the processor for execution;

(iv) repeating steps (i) – (iii) for each instruction in the sequence;

(v) retrieving from said memory a monitoring instruction at the end of the said sequence of instructions that contains a reference value;

(vi) comparing said reference value to the stored monitoring value; and

(viii) confirming whether the proper set of instructions in said sequence have been executed, on the basis of said comparison.

34. (Previously Presented) The method of claim 33, wherein the data contained in the retrieved instruction comprises the numerical value of binary code for the instruction.

35. (Previously Presented) The method of claim 34, wherein the processing of said data comprises a hashing operation performed on said numerical value with the stored monitoring value.

36. (Previously Presented) The method of claim 33 further including the step of retrieving an initial monitoring instruction at the beginning of said sequence that contains an initial monitoring value, and storing said initial monitoring value as the stored monitoring value.

37. (Previously Presented) The method of claim 33, further including the step of interrupting the execution of said program if said comparison indicates that the proper set of instructions has not been executed.

38. (Previously Presented) The method of claim 33 wherein, if said program contains a jump instruction, steps (v) – (vii) are executed prior to said jump instruction.